Large Shared Memory Applications in Visualization

Hansong Zhang, Ph.D.
Silicon Graphics, Inc.

SGI Today

- Core Technology
  - ccNUMA memory controller/network fabric
    - Cache coherent non-uniform memory access
    - Memory coherency protocol runs on NUMALink network fabric
      - 4^{th} generation
      - 3.2GB/s in each direction
    - Itanium2 processor, Linux OS
- Differentiation
  - Large shared memory computing platforms for enterprise, science, and engineering

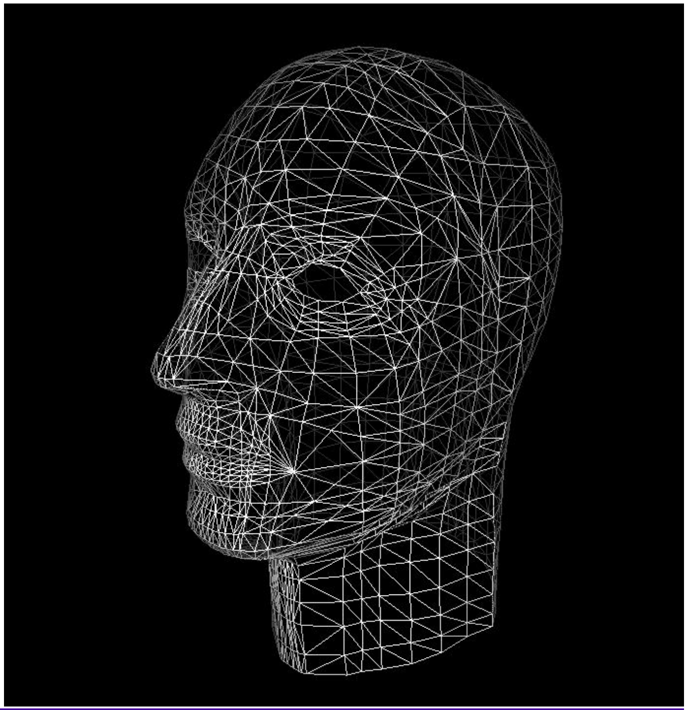
Server-Based Graphics

- Workstation graphics
  - Traditionally, compute and the supercomputer, visualize on the workstation
    - The large data problem
- Server-based graphics
  - Compute on the supercomputer, visualize on the supercomputer, display on the workstation
    - Based on massively parallel graphics algorithms running on CPUs

sgi

Graphics Basics

- Represent the world (Modeling)
  - As surfaces – “surface model”
  - Simplest surface: triangles
  - Approximate all surfaces with triangles
    - What you call a piece-wise linear approximation
- Draw the world (Rendering)
  - Draw triangles



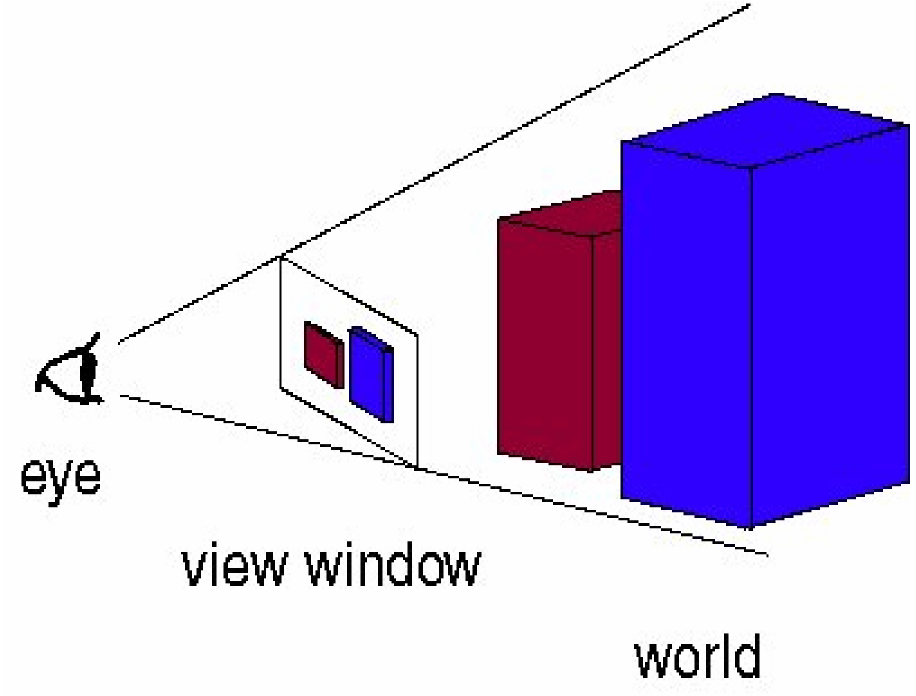
Graphics Basics

- Two fundamental problems in computer graphics
  - **Visibility**: decide which surfaces are visible
    - Visible to the eye?
    - Visible to another surface?
  - **Shading**: determine the look of surfaces
  - Bottom line: simulate light transport
    - The rendering equation

$$L(x,\omega_r) = L_e(x,\omega_r) + \int_{\omega_i \in \Omega} f_r(x,\omega_i \rightarrow \omega_r)\, L_i(x,\omega_i)\, \cos\theta_x \, d\omega_i$$

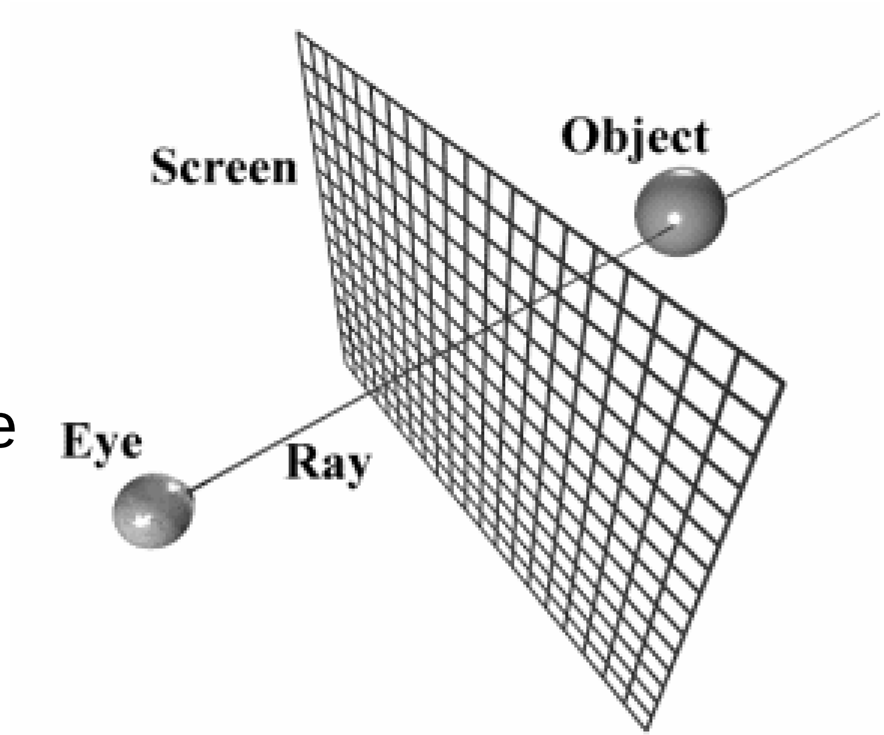
GPUs (Graphics Cards)

- **Go through the model**
  - Project *all* objects onto the screen and pixelize
- Determine the closest surface at each pixel
- Discard projections if they're not closest (a waste)
- The "Last man standing" approach



Ray Tracing

- **Go through the pixels**
- Fire a ray through each pixel
- Find the hit (intersection)
  - Don’t try each surface
  - Spatial data structures guide the ray to “zero-in” on target surface
    - Bounding volume hierarchies
    - Spatial partitioning trees (KD-tree, BSP-tree)



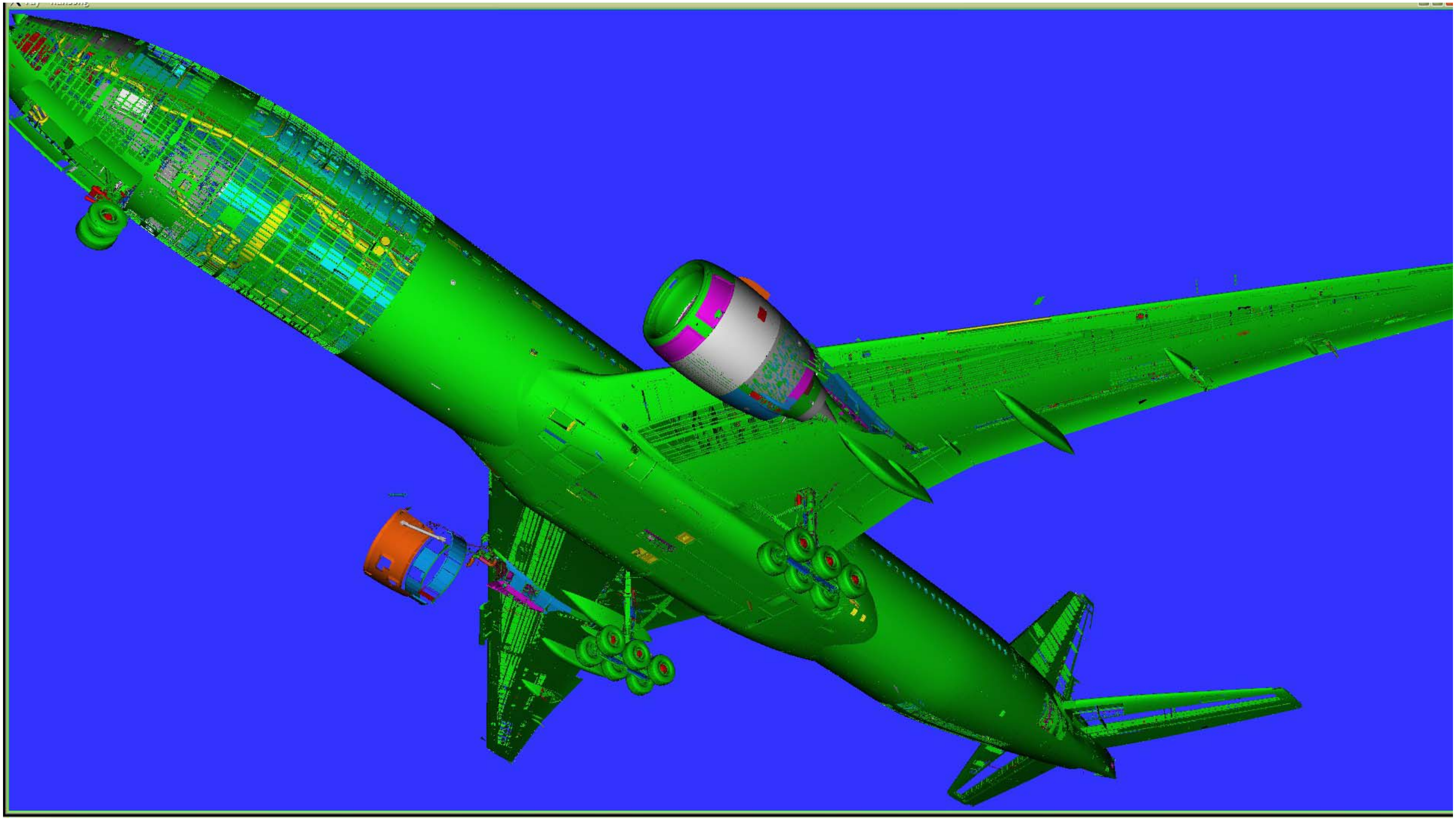
Comparison

- Both can be viewed as a search algorithm
  - Search for the nearest surfaces
- Ray tracing does logarithmic search
  - But data access is semi-random
  - klog(n), with a large k
- GPUs do linear search
  - But data access is sequential
  - Also it takes advantage of “spatial coherency”
- There’s a point where klog(n)<<n even for a large k
  - That’s when the theoretical advantage becomes real

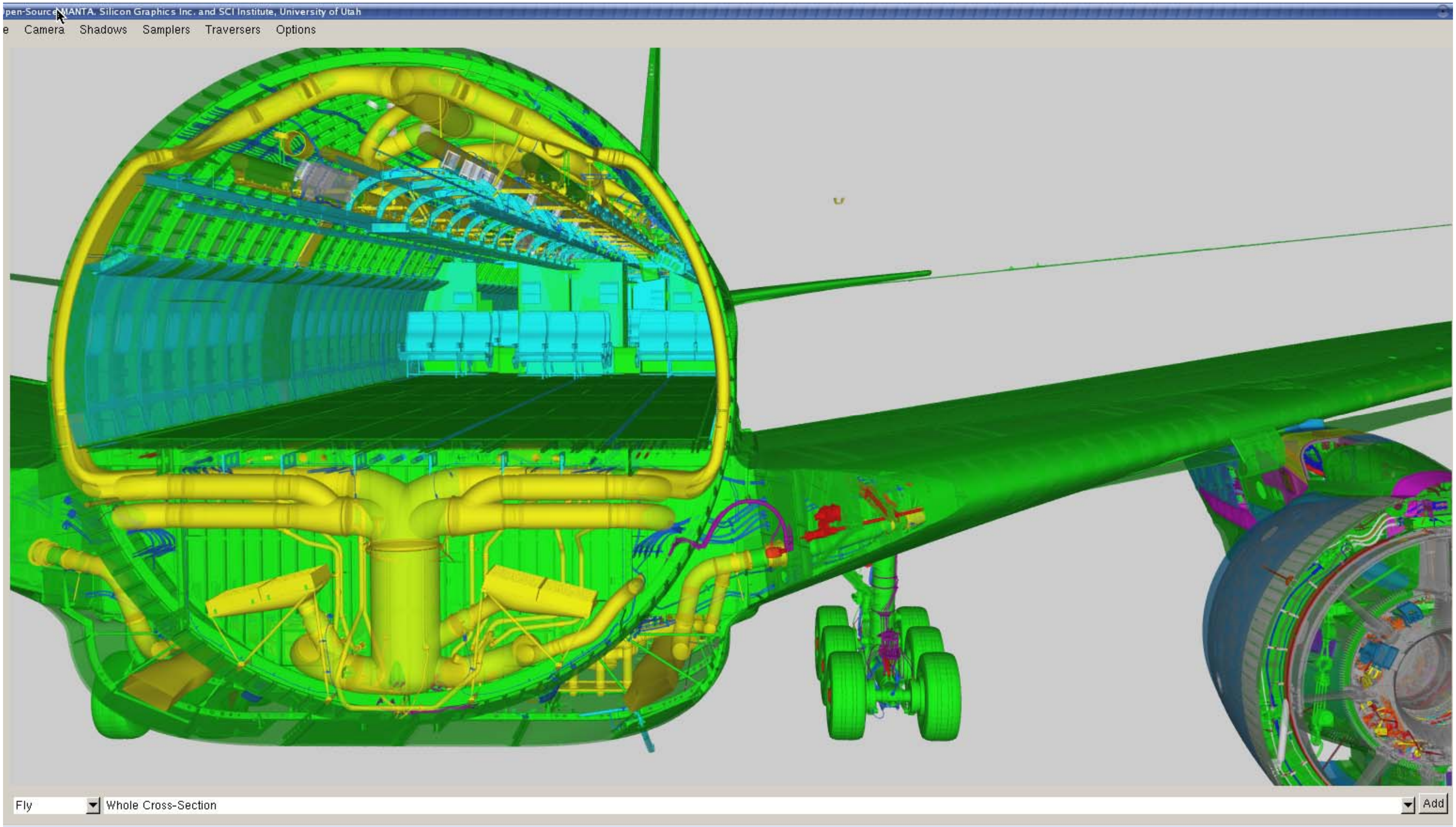
Large CAD Models

- Example: Boeing 777
  - Known for pioneering use of CAD: they modeled the whole plane
  - First 777 within 0.023 inch of perfect alignment
- The model
  - n=**350 million** triangles, 17 GB on disk
  - Exceeds what GPUs can do today

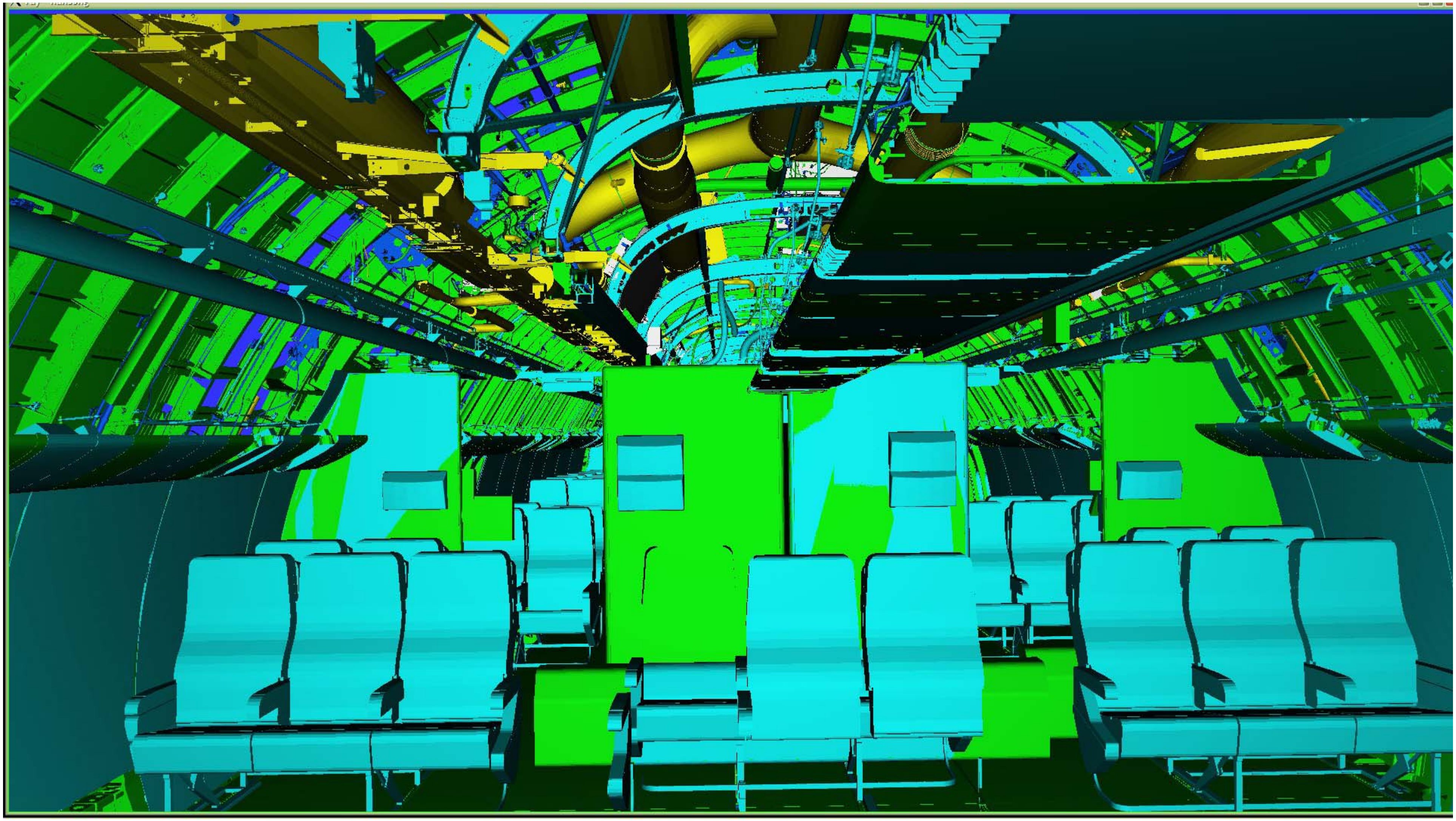
Boeing 777



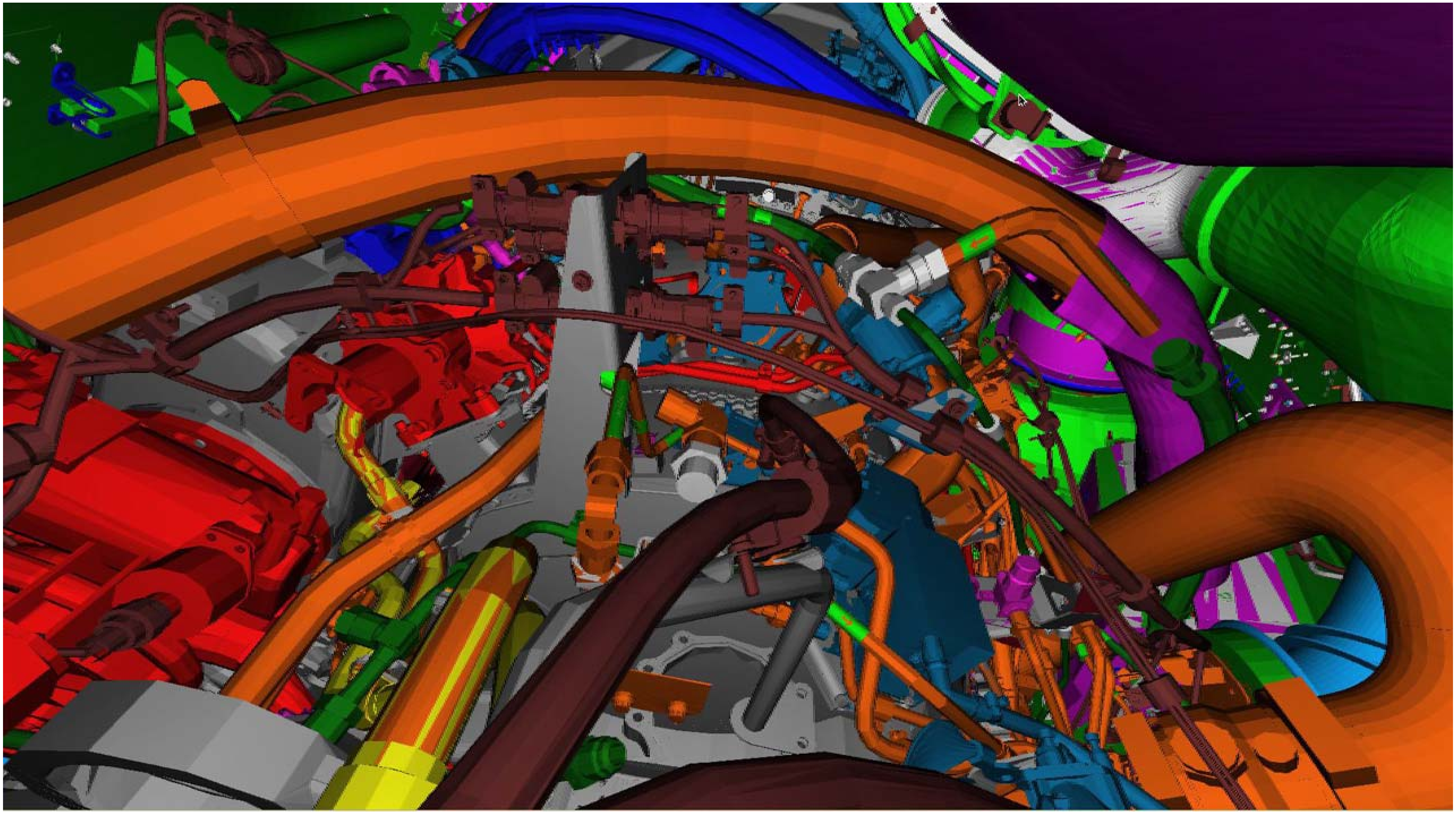
Boeing 777


pen-Source MANTA. Silicon Graphics Inc. and SCI Institute, University of Utah
e Camera Shadows Samplers Traversers Options
Fly
Whole Cross-Section
Add

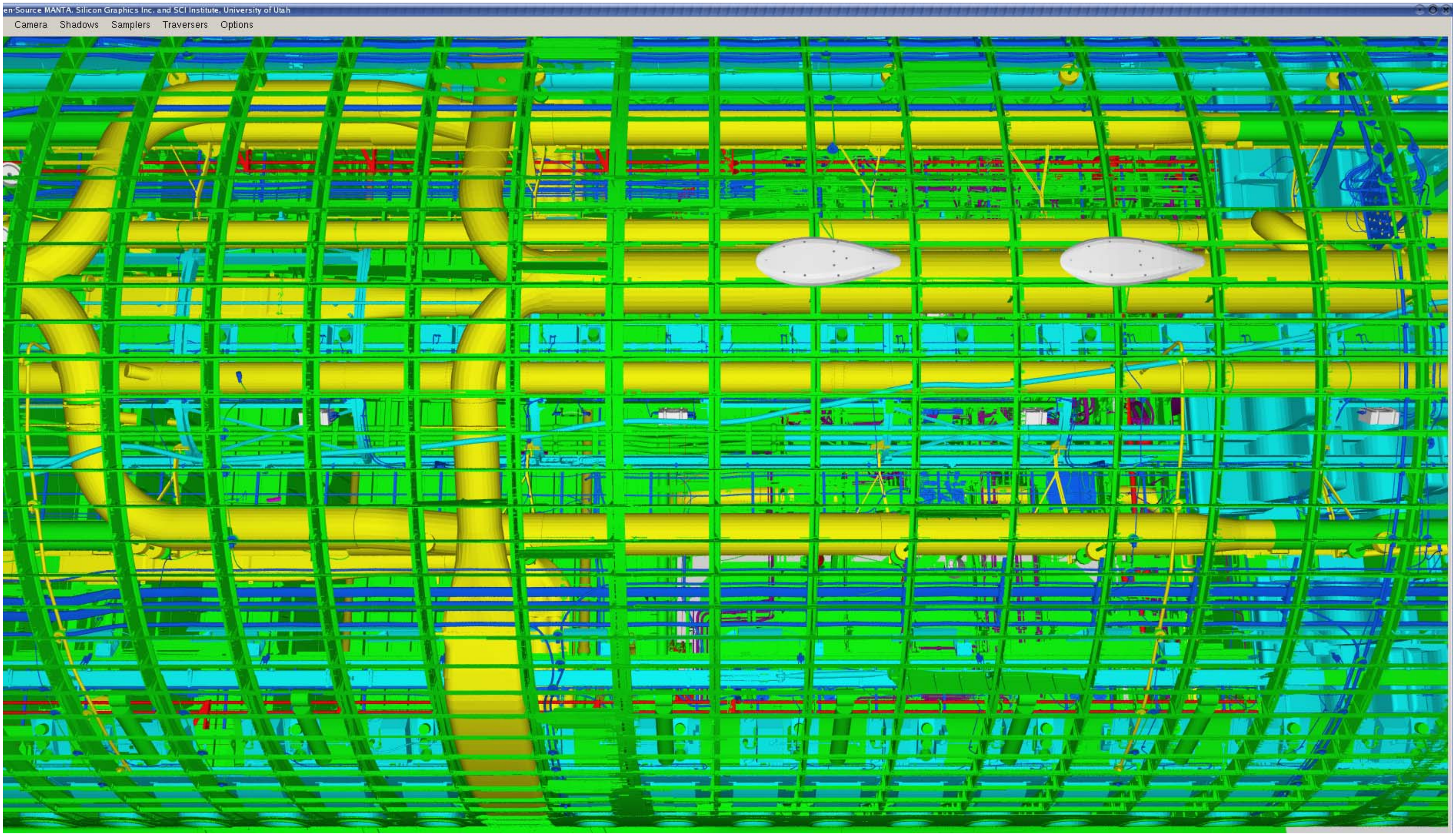
Boeing 777



Boeing 777

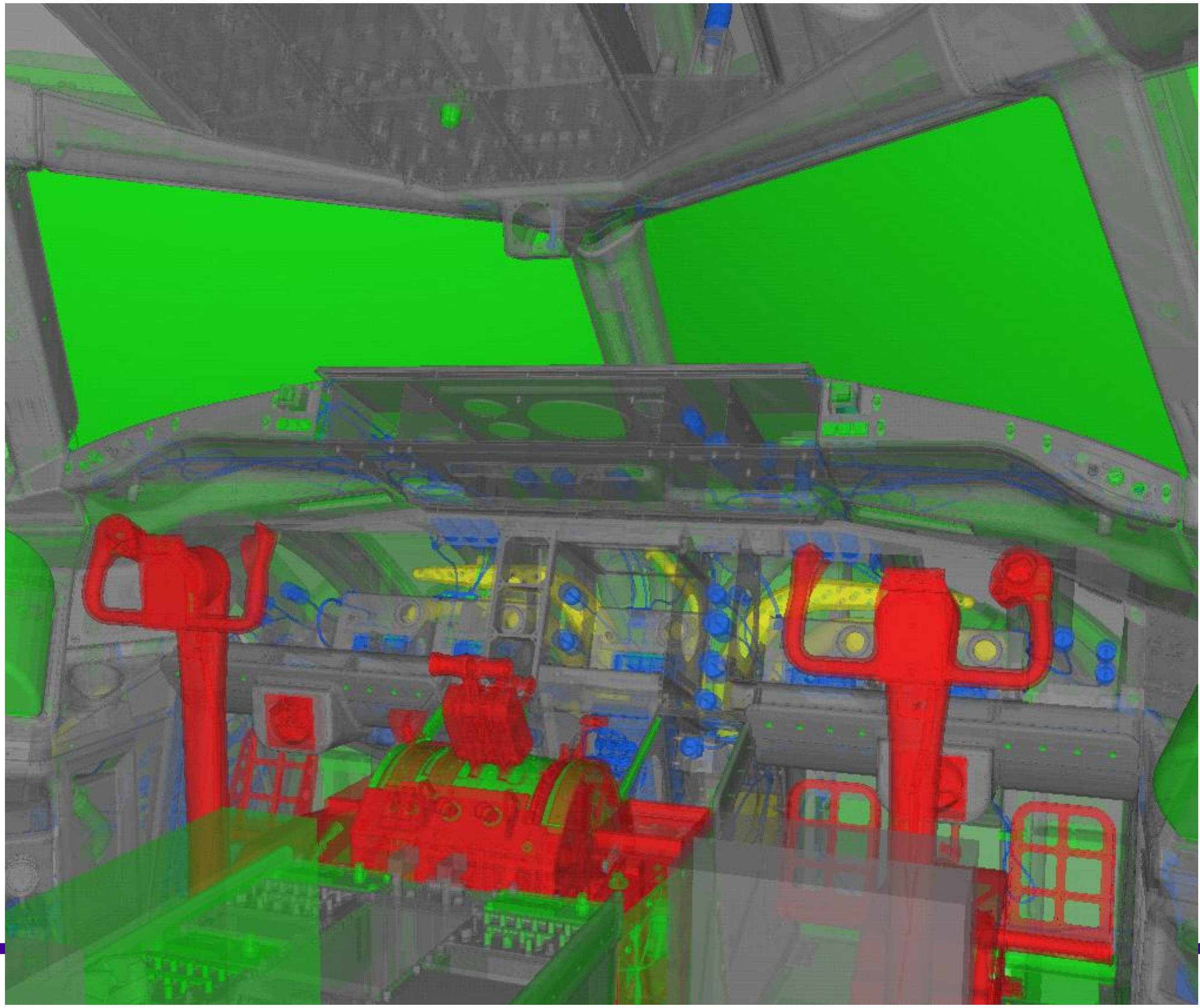


Boeing 777

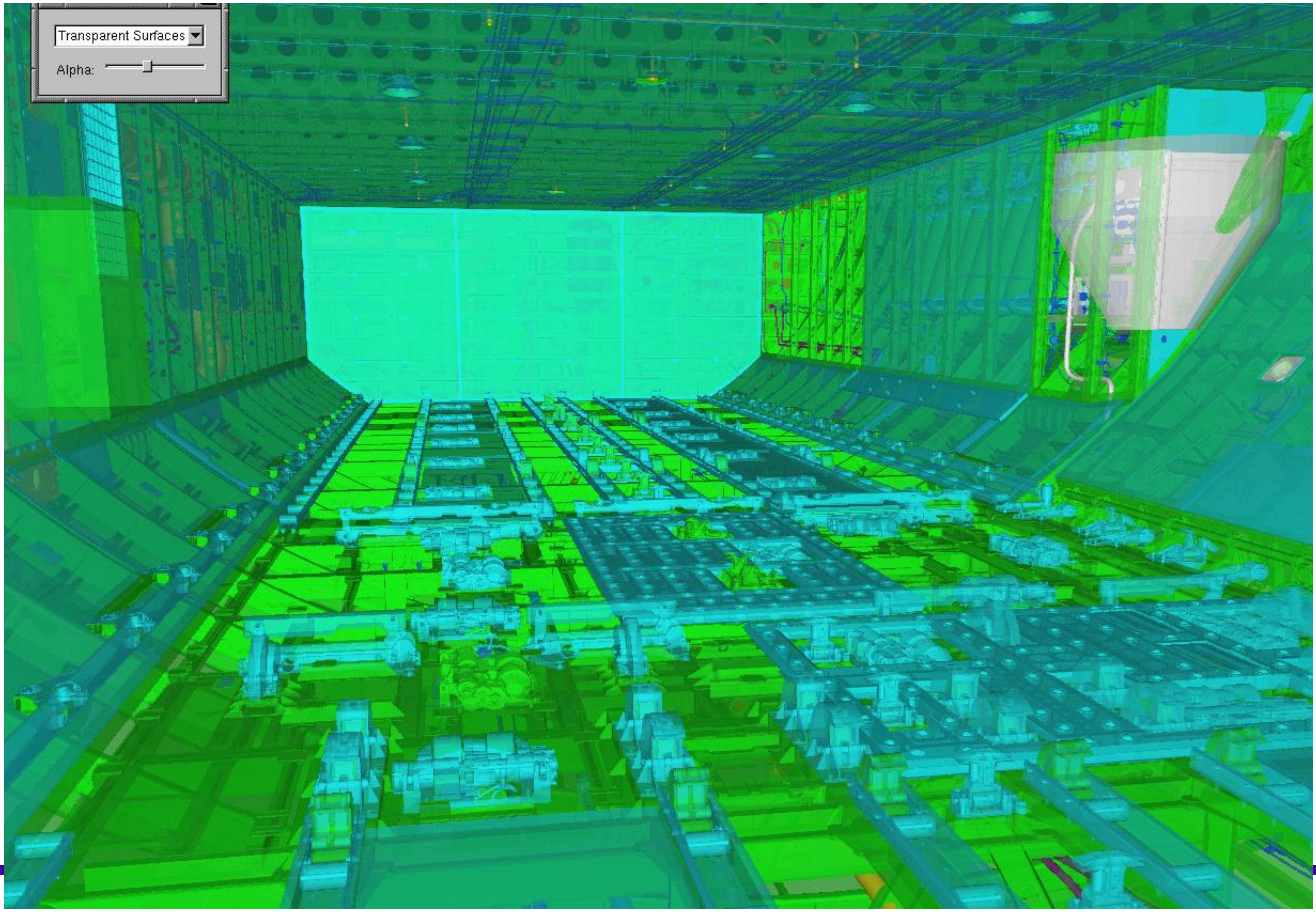


sgi

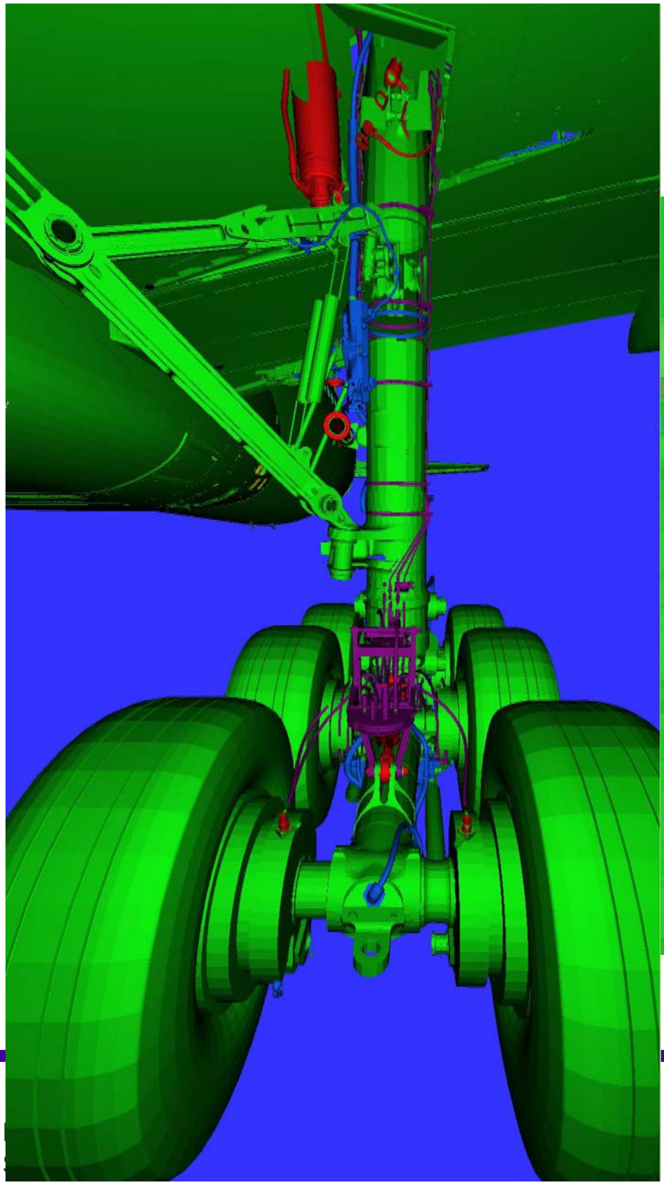
Boeing 777

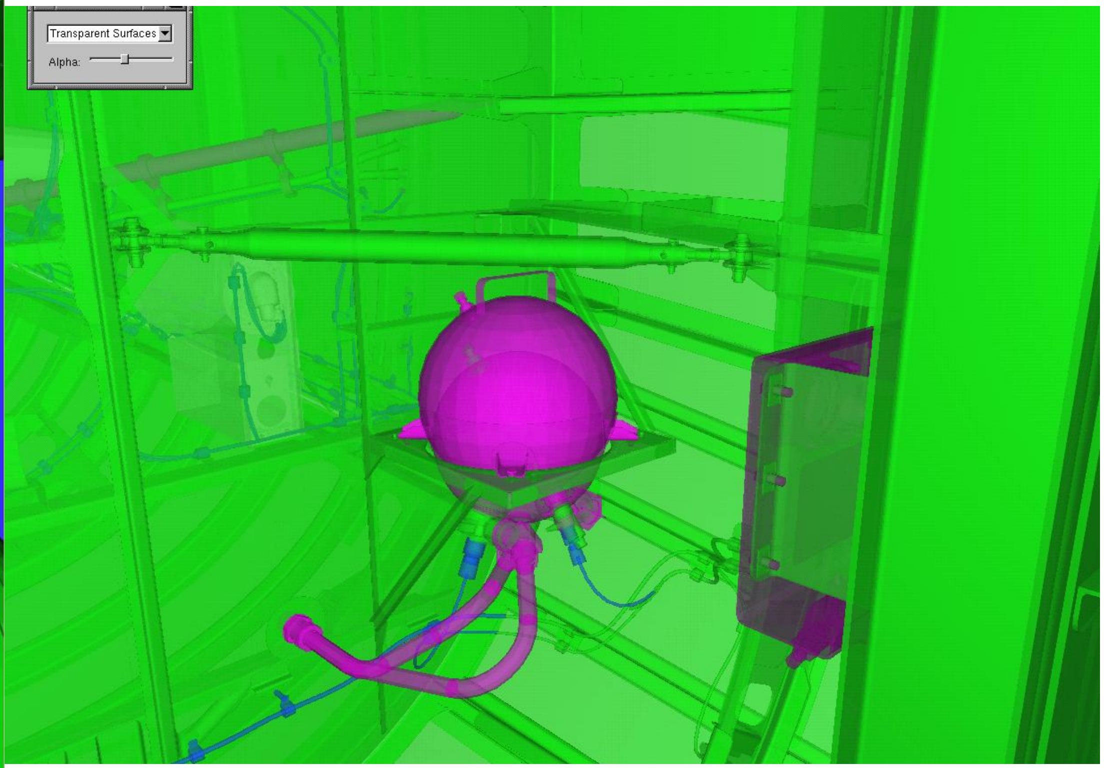


Boeing 777

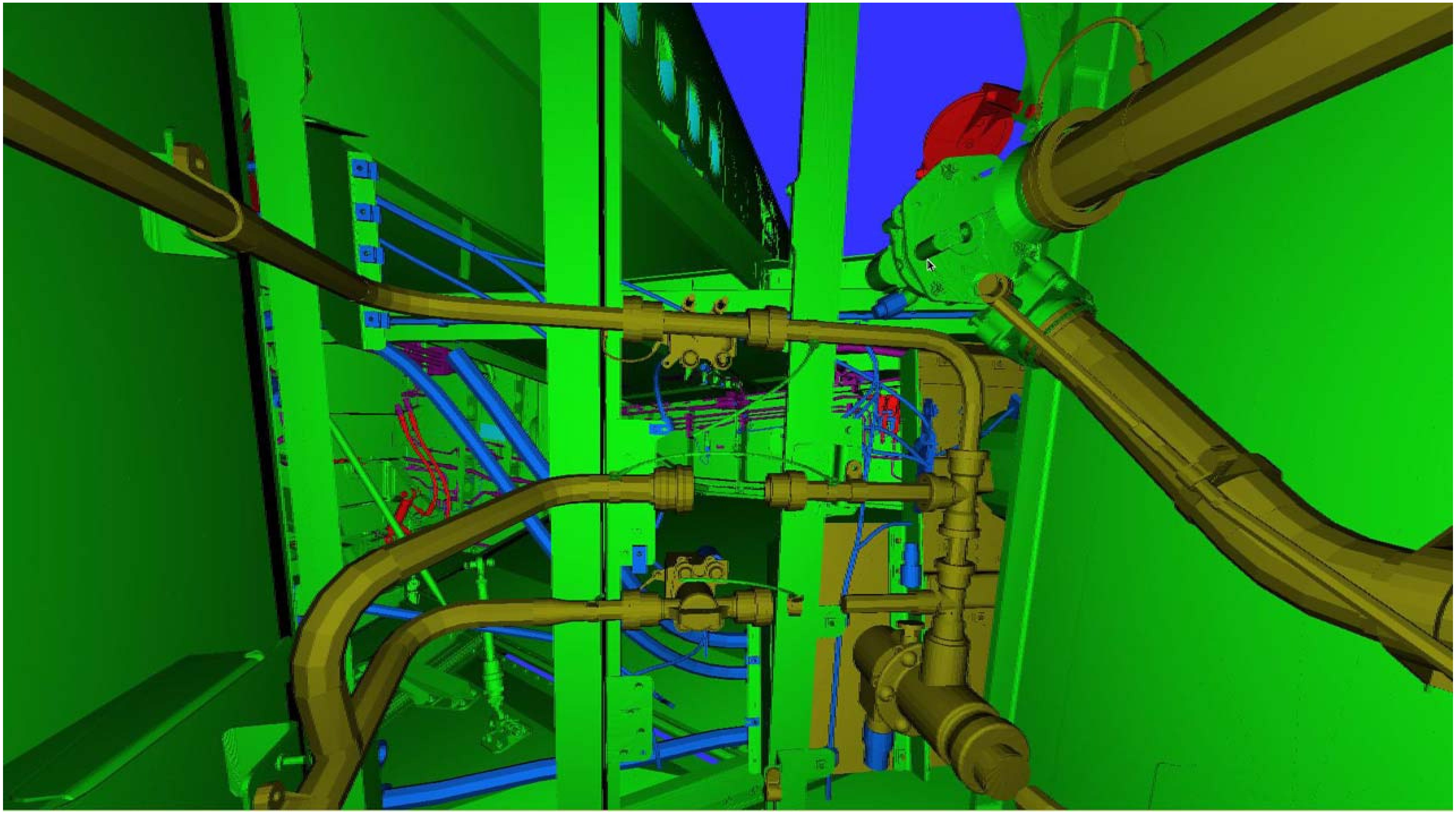


Boeing 777





Boeing 777



sgi

Ray-Tracing Boeing 777

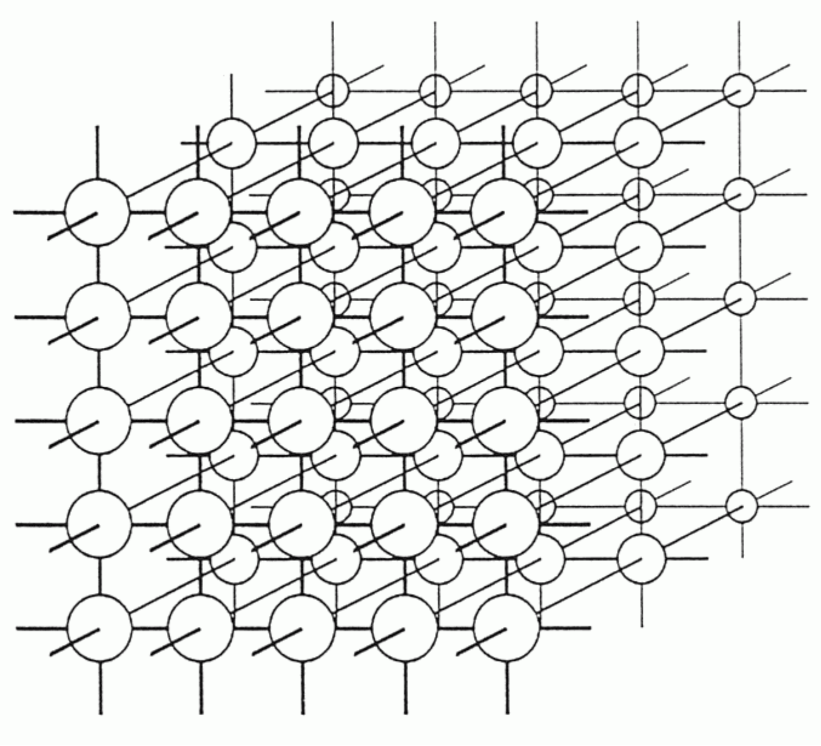
- ~17 GB on disk
- ~17 GB spatial data structures
- 50 GB memory used at run-time
- Average 5 million triangles visible per frame
  - **Finding the right 5 million without touching the rest 345 million** - vital to large data and the key advantage of ray tracing
- Demo'd at Supercomputing 2005 with Boeing and Intel

sgi

Video

Large Volume Data

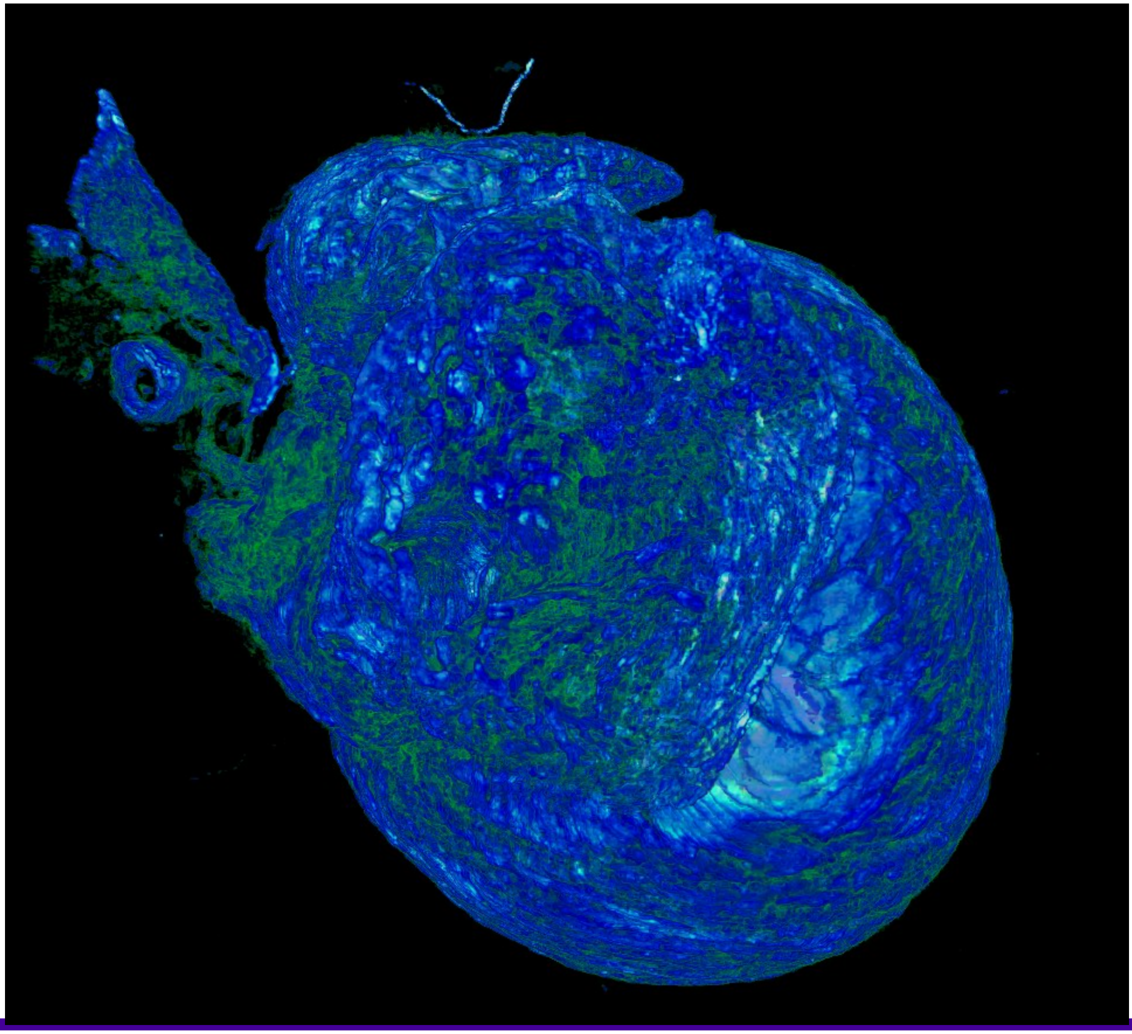
- Data defined on a X-Y-Z grid
- Defines a “field”
- From sensors MRI, CT, seismic/oil-and-gas exploration
- From simulation



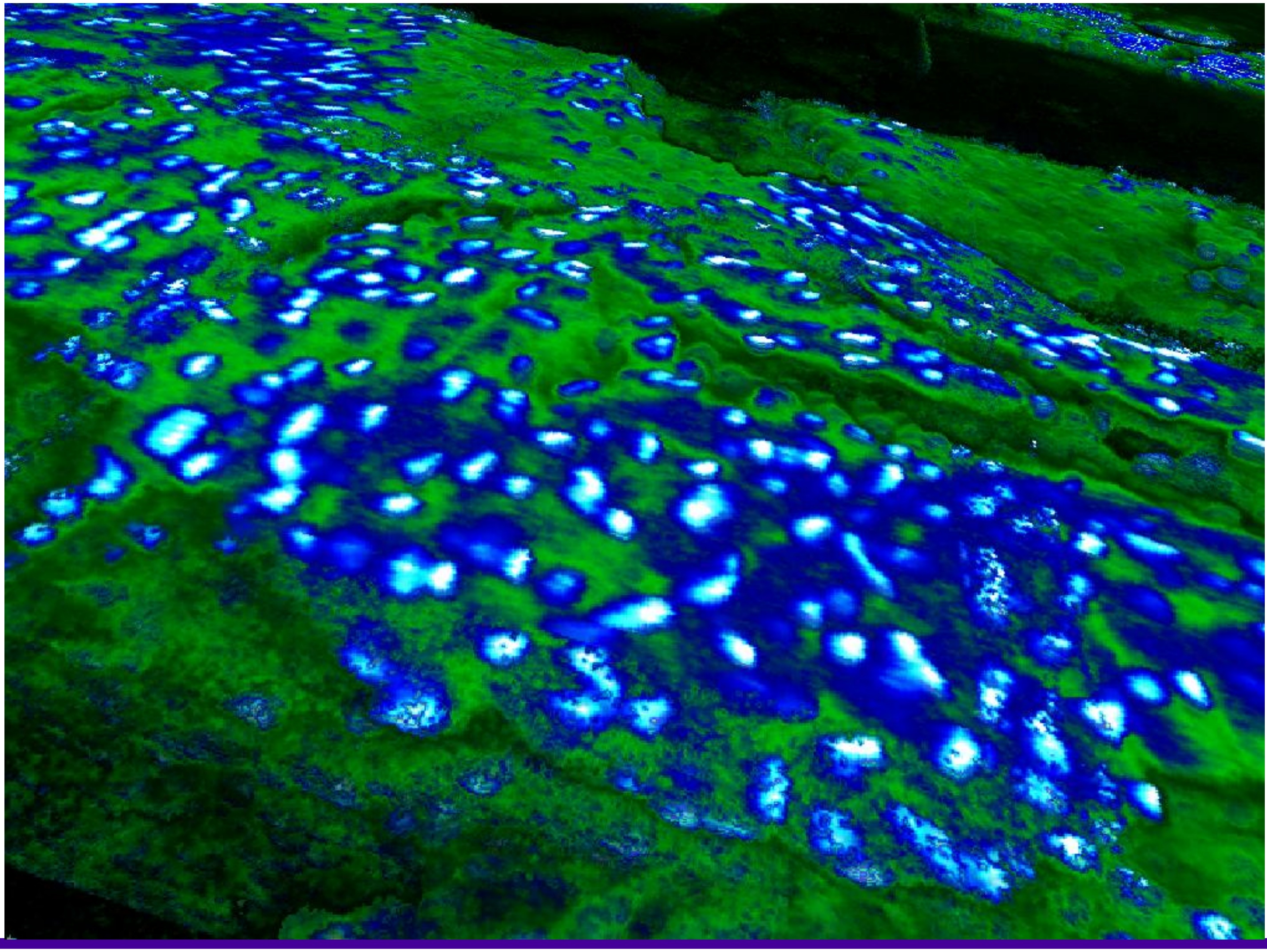
Large Volume Data

- Example: mouse heart dataset from MIT Whitehead BioImaging Institute
  - Microscope resolution: 0.78 micron in X and Y, 2 micron in Z
  - Mouse heart size: 6.37 x 7.27 x 4.2 milimeters
  - Data size: 8167 x 9325 x 2100 x 16-bit per sample x 2 channels
    - 640 GB – per scan of an object of this size

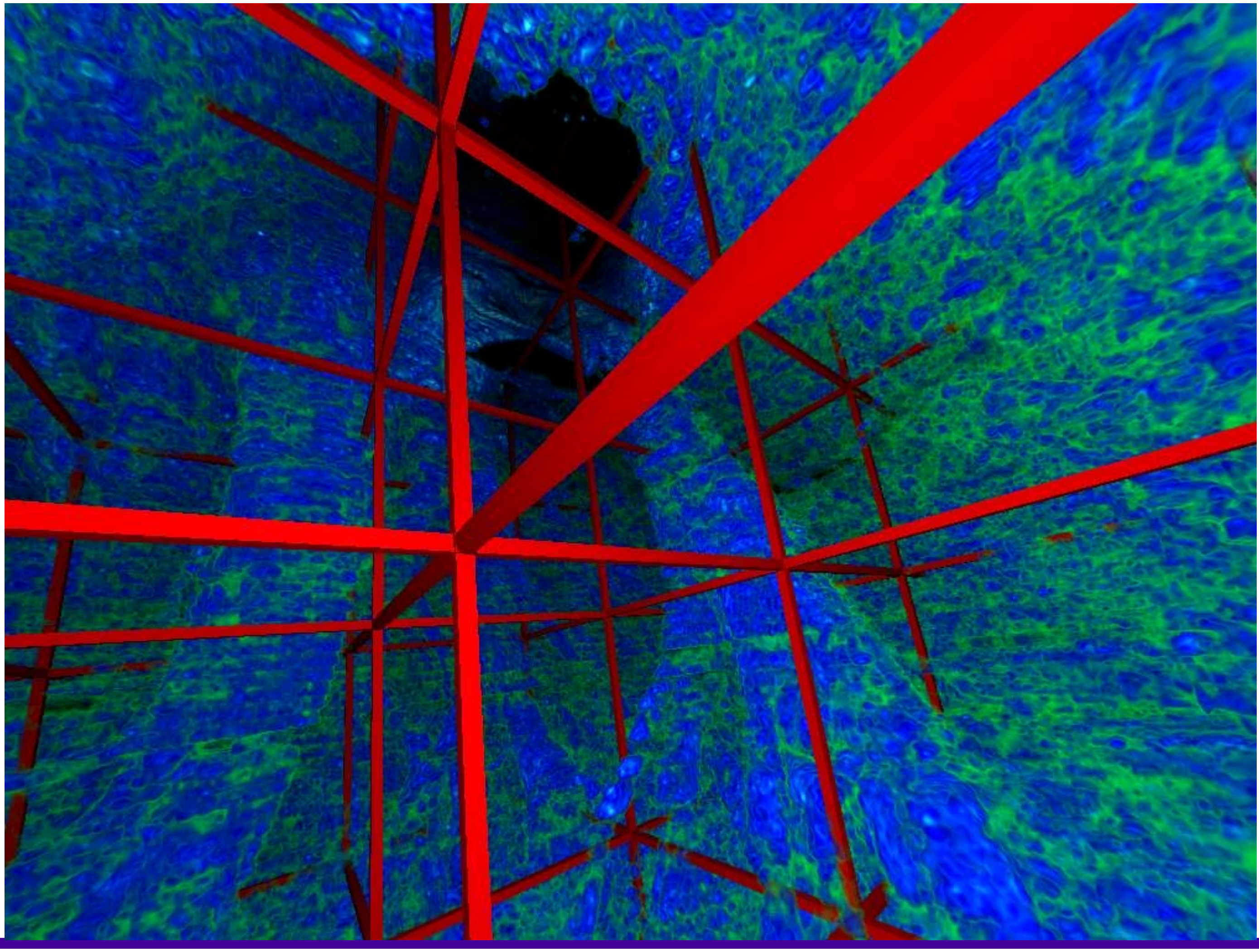
MIT Mouse Heart



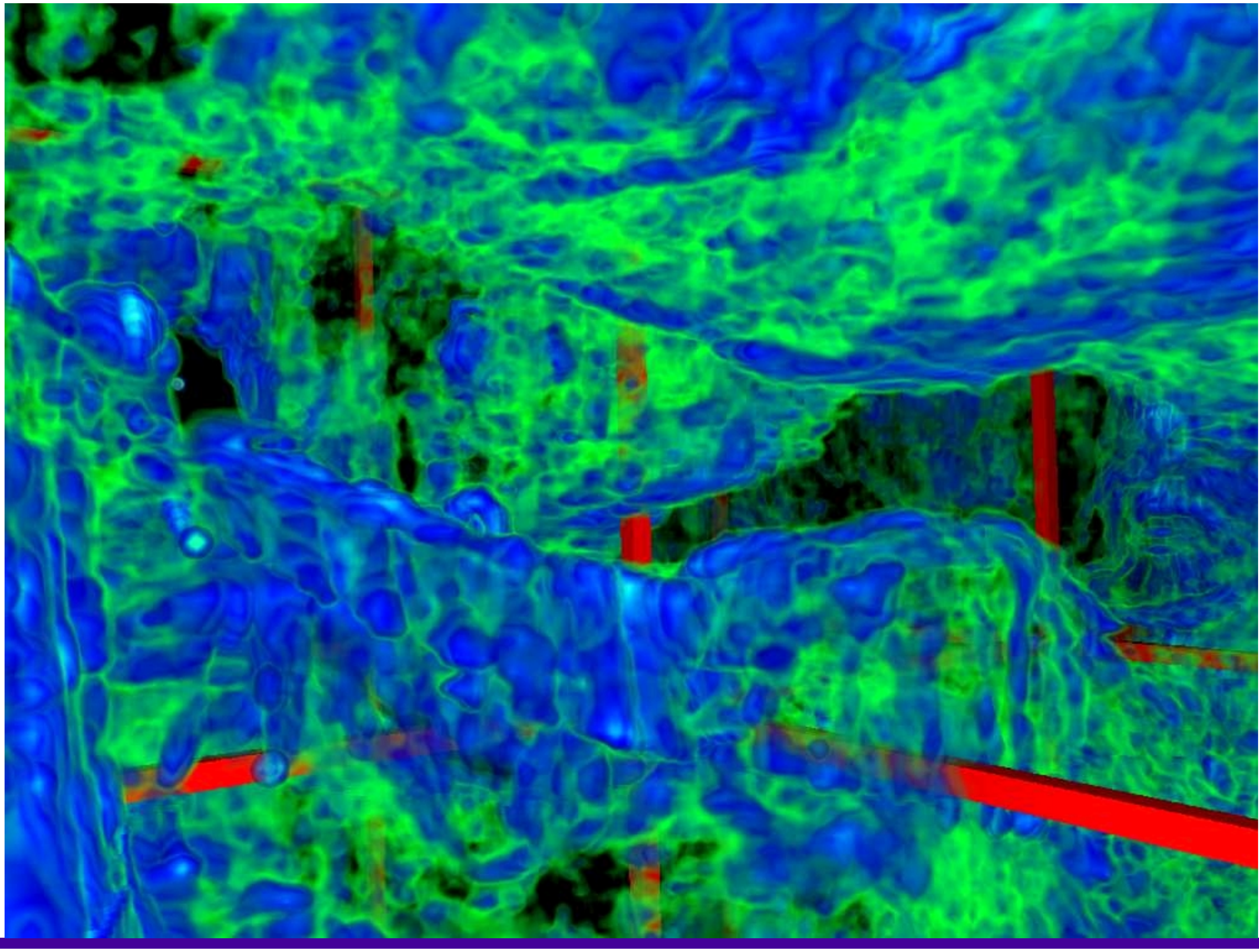
MIT Mouse Heart



MIT Mouse Heart

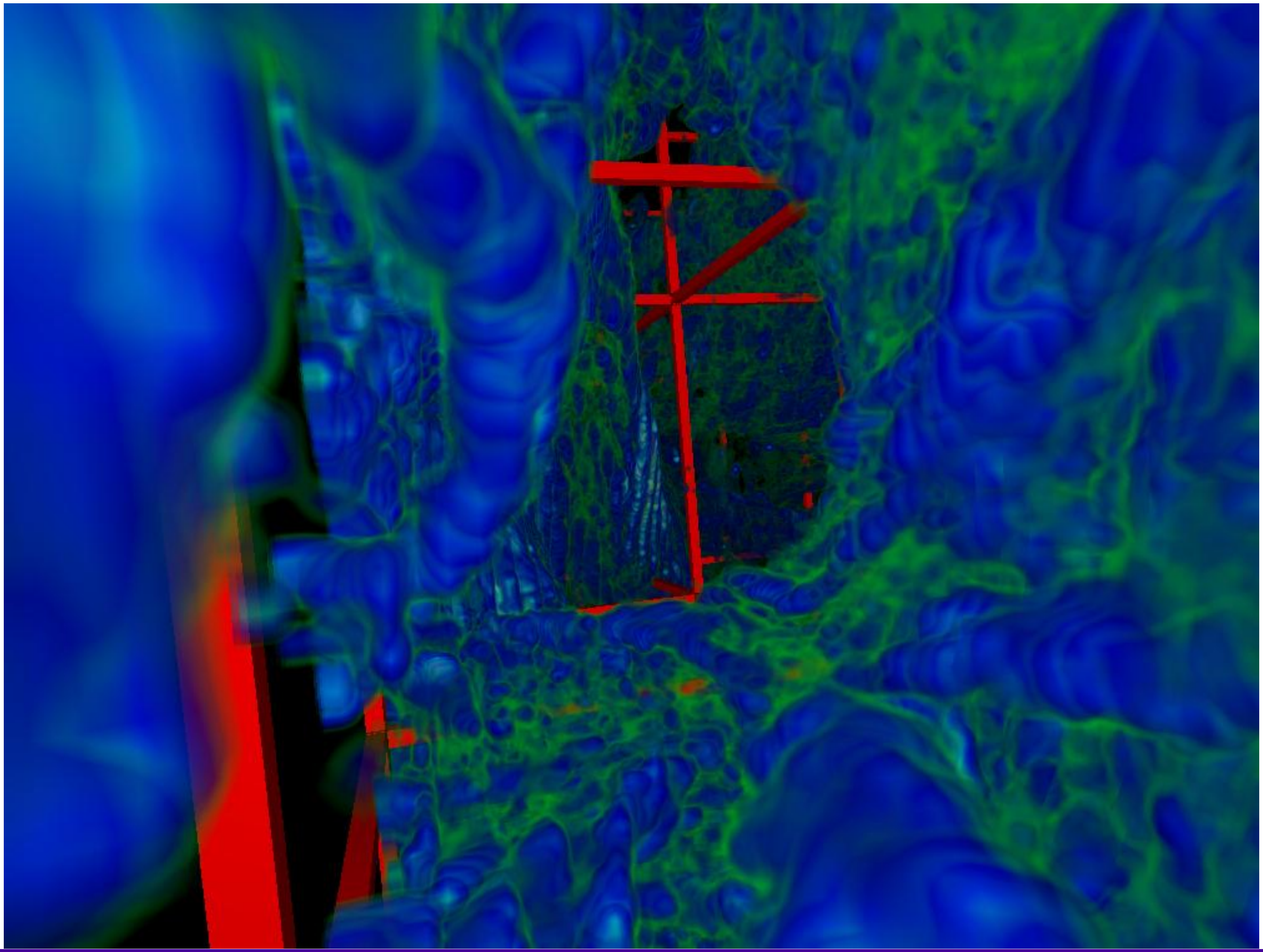


MIT Mouse Heart



sgi

MIT Mouse Heart



Rendering Large Volume Data

- Use ray-tracing
- The key is not to touch data not relevant to generating the current picture
  - Note the theme here
  - Otherwise, the game is lost even before any processing
- Distribute memory carefully
  - Reduce stride (improve cache behavior)
  - Avoid hot spots in memory access
- 64-processor, 256GB system